Encounter Based Sensor Tracking

Andrew Symington
Department of Computer Science
University of Oxford
Oxford, United Kingdom
andrew.symington@cs.ox.ac.uk

Niki Trigoni
Department of Computer Science
University of Oxford
Oxford, United Kingdom
niki.trigoni@cs.ox.ac.uk

ABSTRACT

This paper addresses the problem of tracking a group of mobile sensors in an environment where there is intermittent or no access to a localization service, such as the Global Positioning System. Example applications include tracking personnel underground or animals under dense tree canopies. We assume that each sensor uses inertial, visual or mechanical odometry to measure its relative movement as a series of displacement vectors. Each displacement vector suffers a small quantity of error which compounds, causing the overall accuracy of the positional estimate to decrease with time. The primary contribution of this paper is a novel offline method of counteracting this error by exploiting opportunistic radio encounters between sensors. We fuse encounter information with the displacement vectors to build a graph that models sensor mobility. We show that two dimensional sensor tracking is equivalent to finding an embedding of this graph in the plane. Finally, using radio, inertial and ground truth trace data, we conduct simulations to observe how the number of anchors, transmission range and radio noise affect the performance of the proposed model. We compare these results to those from a competing model in the literature.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Wireless communication

Keywords

Localization, Tracking, Encounters, Rigidity

1. INTRODUCTION

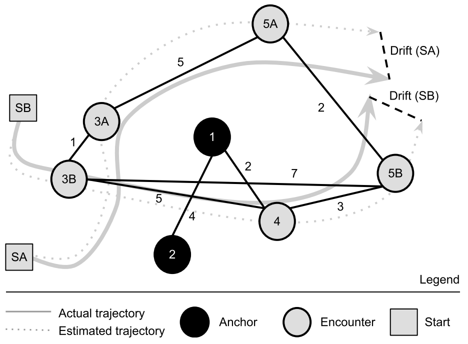
A Wireless Sensor Network (WSN) is a group of interconnected, resource-constrained devices ("sensors") that instrument the environment, store data, perform processing tasks and forward information. There are many applications in which sensor location awareness is advantageous, or even essential. Example applications include tracking mineworkers underground, or groups of animals over long periods.

One widely-used system for localization is the Global Positioning System (GPS). However, many sensor deployments preclude the use of GPS for one or more of the following reasons. Firstly, the service may not be available in satellite-denied environments. Secondly, GPS consumes a significant amount of energy, which reduces the sensors' lifetime. Thirdly, the technology requires expensive hardware, making it infeasible for use in large-scale sensor networks that feature many nodes. GPS is one example of anchor-based localization, which assumes the availability of a set of fixed sensors. Such an infrastructure is often difficult – or impossible – to set up, for example in disaster scenarios.

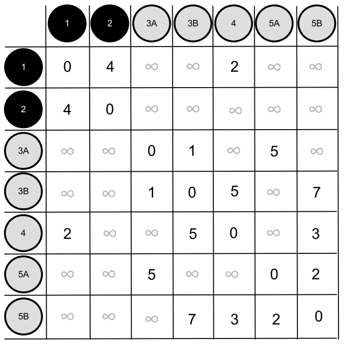
When a fixed localization infrastructure is unavailable, one may measure relative changes in position using an inertial navigation system [9]. The micro electro-mechanical sensing (MEMS) devices used to instrument inertial changes are becoming increasingly energy efficient and cheaper to manufacture, making them feasible for use in WSNs. However, measurements from such devices are typically corrupted by a small quantity of error, arising from bias, noise or information quantization. Since the measured changes are relative, these small errors compound over time. This causes the positional estimate to *drift*, which means that one cannot accurately track a sensor by inertial measurement alone.

In this paper we propose a centralised model that counteracts sensor drift by exploiting opportunistic radio encounters between pairs of sensors, as well as between sensors and anchors. In so doing we provide a method by which sensors may be tracked for longer periods in the absence of a localization infrastructure. We use encounters and displacement estimates to build a graph that models sensor mobility. A vertex represents either the fixed position of an anchor, or the position of a sensor during an encounter. Edges represent the physical distances between vertices. Our proposed tracking model then finds an embedding of this graph in the plane, thereby *localizing* the encounter points. Finally, the original positional estimates are threaded through the embedding to obtain a drift-corrected trajectory. To summarize, our work makes the following contributions:

1. We propose a novel *Encounter Based Tracking* (EBT) model that converts encounters and displacement vectors to a graph, solving the tracking problem with a single application of graph realization.
2. We propose a novel drift correction algorithm, entitled *Radial Drift Correction,* that outperforms *Linear Drift Correction* when used in conjunction with EBT.

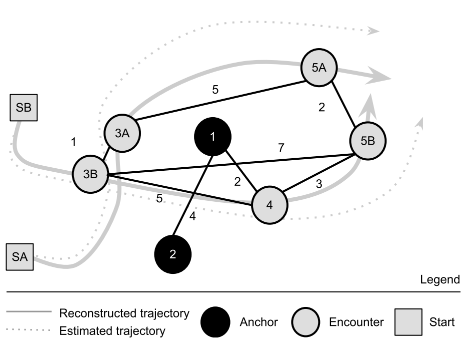


(a) Encounter graph.



	1	2	3A	3B	4	5A	5B
1	0	4	∞	∞	2	∞	∞
2	4	0	∞	∞	∞	∞	∞
3A	∞	∞	0	1	∞	5	∞
3B	∞	∞	1	0	5	∞	7
4	2	∞	∞	5	0	∞	3
5A	∞	∞	5	∞	∞	0	2
5B	∞	∞	∞	7	3	2	0

(b) Distance matrix.



(c) Planar realization of graph.

Figure 1: Overview of Encounter Based Tracking: Diagram 1a shows the actual (solid) and estimated (dotted) trajectories for two sensors, starting at SA and SB. The vertices and edges for the corresponding graph are shown by circles and solid lines respectively. We use displacement vectors drawn from estimated trajectories, as well as distance estimates from radio encounters between sensors and anchors to insert edges into the graph. The positions of the vertices are unknown, and so the graph is described by the distance matrix in 1b. We perform graph realization on this matrix to obtain the planar embedding in 1c. The estimated trajectories are threaded through this embedding to obtain a drift-corrected trajectory for each sensor.

3. We evaluate the performance of EBT using a realistic radio model and trajectories drawn from real trace data. We conduct a sensitivity analysis of EBT and show that it performs well relative to *Directed Diffusion Tracking* [6], a related model from the literature.

The remainder of this paper is organized as follows. Sec. 2 positions our work relative to existing localization literature. Sec. 3 provides an overview of our proposed model. Secs 4 to 6 describe each of our model's three main steps in detail. Sec. 7 discusses the experimental setup used to calibrate our model, and measure its performance. Sec. 8 discusses the results of our experimental study. Finally, Sec. 9 concludes this paper and highlights directions for future work.

2. RELATED WORK

From a sensor networking perspective, localization is typically seen as the process of converting low-level measurements (received signal strength, time of arrival, time difference of arrival) to higher-level information (proximity, range or angle between pairs of sensors) that is then fused together to obtain an estimate of a sensor's location (in relative or global coordinates). Traditional methods localize each sensor in isolation, by estimating multi-hop distances to nearby anchors. Patwari et al [18] define *cooperative localization* as a method whereby the pairwise distances between neighbouring sensors are used to simultaneously localize all sensors in the network. Graph realization is a well-studied [3, 4, 15, 20] family of solutions to the cooperative localization problem, which has seen little application to tracking mobile sensors. Macagnano and de Abreu [17] propose a method of using MDS to track sensors relative to a set of anchor points. However, their representation does not take into account peer to peer encounters, dynamic or kinematic constraints. Cabero et al. [5] propose a Dynamic Weighted Multidimensional Scaling (DWMDS) model for pedestrian tracking that fuses encounters with ranging information. In this model dynamic constraints are encoded directly into a particular graph realization technique. In contrast, our proposed model encodes *measured* displacements directly into the graph itself, allowing a wide variety of graph realization algorithms to be used. The key feature that distinguishes our model is that we integrate mobility information with ranging information to localize sensors simultaneously over space and time. Constandache et al. [6] propose a similar model that uses directed diffusion to propagate position corrections amongst a set of mobile sensors. Their model considers nodes to be collocated during encounters, and thus ranging information is essentially ignored, which we show to significantly affect tracking accuracy.

From a robotics perspective, localization and mapping are dealt with together in the *Simultaneous Localization and Mapping* (SLAM) problem. The objective of SLAM is to fuse process updates (relative changes in the robots pose) with measurement updates (distances to features in the world) in order to find a maximum likelihood estimate (MLE) of the robot's state over time (localization) and the fixed positions of features (mapping). This is closely related to mobile sensor localization, the key exception being that in sensor networks *feature positions and correspondences* are typically known *a priori*. In Thrun and Montemerlo's GraphSLAM [22] a single robot's trajectory is modeled as a graph, which is similar to our model. Subsequently, Kim et al [13] extended GraphSLAM to leverage encounters between robots. Wymeersch et al [23] propose a distributed model called SPAWN that uses a statistical technique, in conjunction with process and measurement models, to infer the state of the sensors over time. Importantly, this model makes the restrictive assumption that sensors move independently according to a memoryless walk. Our approach differs from this family of approaches because we consider the sensor's state as a position only. The 'dissimilarity' between encounters is therefore simply the Euclidean distance, which means that we are able to exploit existing, fast dimensionality reduction algorithms to obtain a solution.

3. ENCOUNTER BASED TRACKING

We begin with the assumption that mobile sensors are equipped with radios, and that they exchange beacons at regular intervals. When a sensor receives a beacon an *encounter* is recorded, which comprises of a time stamp, two sensor identification numbers and an estimated distance. This distance is obtained by passing the signal strength of a radio beacon through a path loss model. We assume further that the sensors use instrumentation in conjunction with a navigation system to determine an *estimated trajectory*. This is essentially a series of two-dimensional displacement vectors, which model the position of the sensor relative to its starting point (0,0). We assume further that this estimated trajectory suffers drift, and so these positional estimates become less accurate with time. At the core of our model lies the intuition that whenever two sensors encounter one another, it relates them in space and time. We model these encounters, and hence sensors' mobility, with a connected graph. The overall goal of EBT is to firstly find an assignment of two dimensional coordinates to the graph vertices (a planar embedding) that satisfies the observed distances, and to then use this embedding to subtract the drift out from the estimated trajectories. EBT does not require anchors to drift-correct the estimate trajectories. However, several anchors (these can be known start or end points on the estimated trajectory) are required to project the final embedding into a usable coordinate frame, such as meters north or east. Fig. 1 provides an overview of EBT, which may be broken down in to the following three steps:

1. *Graph construction* – A graph is built using the encounters and estimated trajectories. Fig. 1a shows the sensors' positions during an encounter as vertices in this graph; edges represent the observed pairwise distances between the vertices. In general, the relative positions (coordinates) of the vertices are not known, and so the problem can be entirely described by the distance matrix given in Fig. 1b.

2. *Graph realization* – This is the process of assigning a two dimensional coordinate to each vertex of the graph, such that the resultant pairwise distances agree with the observed distance matrix in Fig. 1b.

3. *Drift correction* – Each sensor's estimated trajectory is threaded through the embedding. The drift-corrected trajectories are shown as solid lines in Fig. 1c.

The following sections examine each step in closer detail.

4. GRAPH CONSTRUCTION

Fig. 2 shows a toy example that illustrates how the graph is constructed. In this example there are two anchors, shown as dark circles with the numbers 1 and 2. There are two sensors labeled A and B, which follow the path indicated by the corresponding dotted arrows. When these two sensors come within communication range they exchange a radio beacon, and an encounter is recorded. In the example there are two such encounters that occur, depicted as light circles marked 3A/3B and 5A/5B. Two vertices are added for each encounter – one for each of the two sensor positions at that point in time. When encounter is between a sensor and an anchor, only one vertex is added (Circle 4 in Fig. 2).

With the exception of the anchors, the coordinates of the graph vertices are unknown. However, the estimated trajectories, encounters, and anchor positions (if available) provide us with information about the pairwise distances between vertices. In our model this information is captured by graph edges. We distinguish between three edge types:

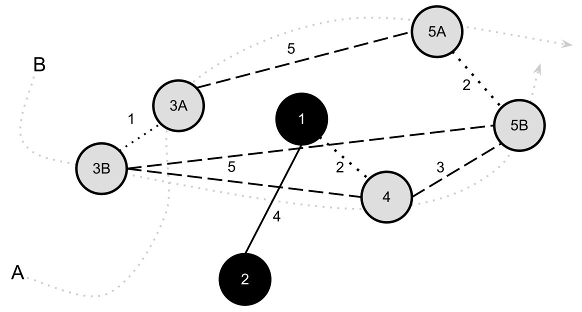


Figure 2: Sensor and anchor positions are shown by light and dark circles respectively. The dotted arrows show the trajectories of sensors A and B. Circles and straight lines correspond to graph vertices and edges respectively. Dashed, dotted and solid edge lengths are obtained respectively from estimated trajectories, the signal strength of a radio beacon, and the known distances between anchors.

1. *Mobility edges* – Recall that encounters occur at discrete points in time. The distance between two encounters on a single sensor's trajectory is given by sampling that sensor's estimated trajectory over the appropriate time interval. Mobility edges are shown by dashed lines in Fig. 2.

2. *Anchor edges* – These are included only when anchor information is available. Since the position of each anchor is known *a priori*, it follows that the subgraph for the anchors is fully-connected. Anchor edges are shown by solid lines in Fig. 2.

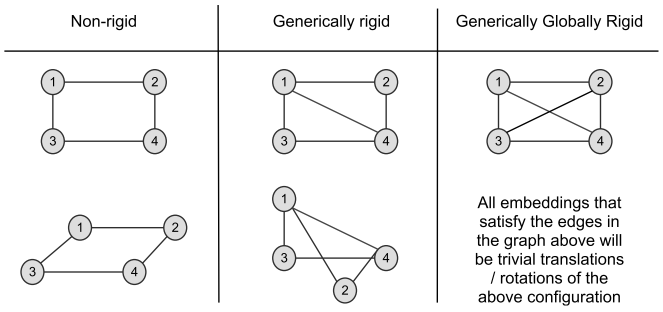
3. *Radio edges* - The distance separating two sensors, or sensor and an anchor, is found using a path loss model, which estimates a distance between the two vertices from the signal strength of the received beacon frame. Radio edges are shown by dotted lines in Fig. 2.

4.1 Vertex merging

In Fig. 2 the layout of the graph is relatively simple, as there are very few encounters. Depending on the radio communication range and beaconing rate, a series of duplicate encounters may arise from sensors being spatially and temporally collocated for short periods. Since the complexity of EBT scales with the number of graph vertices, this number should be kept to a minimum. To this end, we propose the following vertex merging algorithm. We begin by selecting some constant time threshold T_m, which defines the time granularity of our graph. The merging algorithm searches for a pair of duplicate encounters with the smallest time difference. If this time difference is less than T_m, both encounters are *merged* into a single encounter, with a time and distance equal to the mean of the merged pair. The algorithm then repeats until no matching pair is found.

4.2 Edge selection

Consider the subgraph that contains only the vertices for a single sensor (eg. vertices 3B, 4 and 5B form an example subgraph for Sensor B in Fig. 2). Such a subgraph



Non-rigid
Generically rigid
Generically Globally Rigid
1
2
3
4
All embeddings that satisfy the edges in the graph above will be trivial translations / rotations of the above configuration

Figure 3: Left, a flexible graph that fails generic local rigidity: a continuous force applied to vertices 1 and 2 shifts them relative to vertices 3 and 4. Middle, a locally rigid graph structure that is not globally rigid: vertices are rigid against a small continuous force, there exists a reflection (edge $e_{1,4}$) that satisfies the observed edges. Right, a globally rigid graph with a unique planar embedding.

forms a crude, discrete time trace of the given sensor's trajectory. We are able to measure the distance between any pair of vertices in this subgraph by sampling the estimated trajectory. We therefore have sufficient information to fully-connect this subgraph. We refer to this as the *complete edge selection rule*. This rule always yields a subgraph that can be uniquely embedded in the plane.

As a result of drift, a displacement vector measured between two points on an estimated trajectory contains a quantity of error, which increases proportionally to the time separating the two points. An alternate approach to the *complete edge selection rule* is to remove those edges connecting vertices with a large time differences. However, we should only remove those vertices that retain the subgraph's unique embeddability. To achieve this, we must first introduce the notion of graph rigidity [8]. We will use this theory to derive a *conservative edge selection rule*, that seeks to connect the graph with a minimum number of short edges in a way that preserves unique planar embeddability.

We begin by considering only *generic graphs* – those having no $d + 1$ vertices that are collinear in $\mathbb{R}^d$. Such graphs are said to be *rigid* if they do not bend or flex under a small force. Generally speaking, the more edges one adds to a graph the more rigid the graph becomes, as a result of more constraints being placed on motion. A graph containing n vertices has $2n$ possible independent motions in two dimensions, corresponding to a horizontal and vertical translation of each node. It is impossible to constrain the three global graph motions, namely a horizontal translation, vertical translation and rotation. This intuitive notion of constrained movement lies at the core of Theorem 1 below, which provides a test for two dimensional rigidity [16].

THEOREM 1. *The edges of a graph $G = (V, E)$ are independent in $\mathbb{R}^2$ iff. no subgraph $G' = (V', E')$ has more than $2n' - 3$ edges, where n' is the number of vertices in G'.*

Graphs that satisfy the Laman condition are referred to as being *locally rigid*. Furthermore, a graph is said to be *redundantly rigid* if and only if the removal of any single edge results in a graph that is locally rigid. Local and redundant rigidity do not necessarily imply unique graph realizability. Graphs that satisfy a stronger condition, *global rigidity*, are rigid against both continuous and discontinuous forces. Refer to Fig. 3 for examples of flexible and rigid graphs.

Theorem 2 frames global rigidity certification as composition of 3-connectivity[1] and redundant rigidity testing, both having polynomial time algorithms. Refer to Jackson and Jòrdan [10] for a proof of this theorem. No equivalent result exists for three dimensions or greater.

THEOREM 2. *A graph $G = (V, E)$ with $n \geq 4$ vertices is generically globally rigid in $\mathbb{R}^2$ if and only if it is 3-connected and redundantly rigid in $\mathbb{R}^2$.*

Having outlined conditions for unique localizability, we may now define the *conservative edge selection rule*. Consider the case where we connect every vertex i to its three neighbours, as shown in Fig. 4. The resulting graph is an instance of a trilateration graph [24] and is therefore globally rigid. Applying this rule over the entire trajectory results in the subgraph being globally rigid. However, this rule does not guarantee global rigidity of the entire graph.

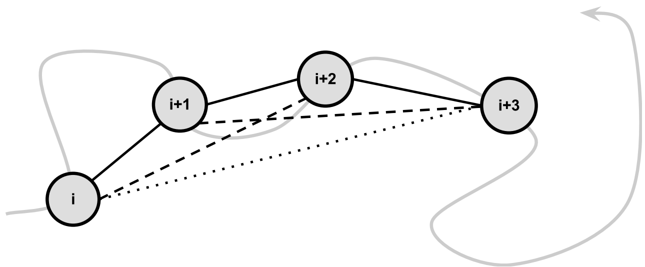


Figure 4: Dotted lines show a globally rigid subgraph formed by connecting each vertex to its three adjacent neighbours. If all encounters are connected in this way, the resultant graph is globally rigid.

5. GRAPH REALIZATION

Consider a graph $G = (V, D)$ with V being a set of n vertices, and D being a $n \times n$ symmetric matrix of pairwise distances between these vertices. The *graph realization problem* seeks to find a d-dimensional embedding – the coordinates of V in $\mathbb{R}^d$ – that preserves D. The d-dimensional graph realization problem is an instance of dimensionality reduction that is provably NP-Hard in the general case [19], and also when radio propagation is modelled as a unit disc graph [2]. We showed in the previous section that if one assumes that the graph vertices are algebraically independent, then it is possible to certify the uniqueness of a planar embedding. The embedding itself may be found using one of many graph realization algorithms.

In this section we survey three types of graph realization that are commonly used for static sensor localization. We conclude this section with an analysis of their applicability for use in our proposed Encounter Based Tracking model.

[1]A graph $G = (V, E)$ with n vertices is *k-connected* if the removal of any $i < k$ vertices results in a connected subgraph $G' = (V', E')$ with $n - i$ vertices.

5.1 Multidimensional Scaling (MDS)

The objective of *multidimensional scaling (MDS)* is to find an embedding $X = \begin{bmatrix} \mathbf{x}_1 & \cdots & \mathbf{x}_n \end{bmatrix}^T$ of all n vertices in d-dimensional space, given some distance matrix D, through the minimization of the stress function shown in Eqn 1.

$$X = \arg\min_X \sum_{<i,j>\in D} \left(\left\| \mathbf{x}_i - \mathbf{x}_j \right\| - d_{ij} \right)^2 \tag{1}$$

A shortcoming of MDS is that a complete distance matrix is required to find a solution, which is typically unavailable. Typically, unknown edges are calculated using a shortest path method, as in MDS-MAP by Shang et al. [21].

MDS has the side-effect of pulling nodes towards the centroid of the embedding, meaning that it tends to perform poorly on irregularly-shaped graphs. Shang and Ruml [20] proposed an extension, called MDS-MAP(P), in which MDS-MAP is applied in patches across the network. A patch is a subgraph produced for every node in the network, including all neighbours within some hop count (typically 2-hops). For each patch an MDS solution is first obtained, which is then followed by an iterative least squares minimization of stress between the measured and MDS distance values. The maps are then merged together to form a global map.

5.2 Spectral Graph Drawing (SGD)

Like MDS, spectral graph drawing (SGD) may be expressed as a minimization problem [15]. The objective is to find a planar realization of n vertices. The *neighbourhood* $N(i)$ is a set of vertices connected by an edge to vertex i. Eqn 2 defines W as a *weighting matrix* derived from distances, K as the *degree matrix* and L as the *Laplacian matrix*.

$$[W]_{ij} = w_{ij}, \quad [K]_{ii} = \sum_{j=N(i)} w_{ij}, \quad L = K - W \tag{2}$$

SGD finds an optimal embedding $\tilde{\mathbf{x}}$ that satisfies the minimization problem shown in Eqn 3. It therefore favours solutions that push together those vertices connected by high weighted edges. Clearly, the edge weights should be inversely proportional[2] to distance. One trivial solution that satisfies this minimization problem is to select all nodes to be at the same position. To prevent this, a constraint $\mathbf{x}^T\mathbf{x} = 1$ is added, which forces the solution to have a non-zero variance. The selection of the variance to be $1/n$ is arbitrary and simply governs the spread of the solution. The second constraint $\mathbf{x}^T\mathbf{1}_n$ sets the mean of the solution to zero, making the solution translation-invariant.

$$\begin{aligned} \tilde{\mathbf{x}} &= \arg\min_{\mathbf{x}} \left[\mathbf{x}^T L \mathbf{x} \right] \\ s.t \quad & \mathbf{x}^T\mathbf{x} = 1 \\ & \mathbf{x}^T\mathbf{1}_n = 0 \end{aligned} \tag{3}$$

Koren [15] argues that SGD performs poorly on irregular graphs, as it tends to draw nodes into the graph centroid. Koren goes on to propose a variant of SGD, called *degree normalized spectral graph drawing* (DN-SGD). This model updates the variance constraint of SGD to weight the position of the graph vertices by their degree. That is, the constraint $\mathbf{x}^T\mathbf{x} = 1$ in Eqn 3 is replaced by $\mathbf{x}^T K\mathbf{x} = 1$. For regular graphs, the degree matrix is close to a scaled identity matrix, which makes the constraint optimization problem equivalent to that of regular SGD. One of Koren's primary contributions is a proof showing that the solution to this constrained optimization problem is equivalent to finding the generalized eigenvectors of (L, K). Broxton et al. [4] applied DN-SGD to static localization and found that it required a refinement step to obtain a reasonable embedding.

[2]In spectral graph drawing the edge weighting $w_{ij} \in W$ is typically calculated as either $\frac{1}{1+d_{ij}}$ or $e^{-d_{ij}}$ for $d_{ij} \in D$.

5.3 Semidefinite Programming (SDP)

Assume that a graph contains n nodes and m anchors, with known positions $A = \{a_1, \ldots, a_m\}$. Let $D \in \mathbb{R}^{n\times n}$ represent the pairwise distances between all nodes. Similarly, let $\overline{D} \in \mathbb{R}^{m\times n}$ represent the pairwise distances between all nodes and anchors. A value of zero in either $d_{ij} \in D$ or $\bar{d}_{ij} \in \bar{D}$ indicates that no distance information is available. The sets $N_x = \{(i,j) : \forall_{i<j} d_{ij} \neq 0\}$ and $N_x = \{(k,j) : \bar{d}_{kj} \neq 0\}$ therefore contain all unique node-node and node-anchor edges respectively. Semidefinite programming (SDP) is a method of solving a minimization problem with a specific form, using a interior point algorithm. The sensor localization problem may be expressed as the minimization problem Eqn 4, with $X = \{x_1, \ldots, x_n\}$ being a candidate embedding, and $\tilde{X}$ being the optimal embedding. The α and β terms are slack variables that seek to minimize the error between the observed and the predicted distances. Although we have not included it in this paper for space reasons, this formulation of the sensor localization problem may be relaxed into SDP form.

$$\begin{aligned} \tilde{X} \quad &= \quad \arg\min_X \left[\sum_{(i,j)\in N_x} (\alpha^+_{ij} + \alpha^-_{ij}) + \sum_{(k,j)\in N_a} (\beta^+_{kj} + \beta^-_{kj}) \right] \\ s.t \quad & \|x_i - x_j\|^2 + \alpha^+_{ij} - \alpha^-_{ij} = d^2_{ij} \quad \forall_{(i,j)} \in N_x \\ & \|x_i - a_j\|^2 + \beta^+_{kj} - \beta^-_{kj} = \bar{d}^2_{kj} \quad \forall_{(k,j)} \in N_a \\ & \alpha^+_{ij} \geq 0, \quad \alpha^-_{ij} \geq 0, \quad \forall_{(i,j)} \in N_x && (4) \\ & \beta^+_{kj} \geq 0, \quad \beta^-_{kj} \geq 0, \quad \forall_{(k,j)} \in N_a && (5) \end{aligned}$$

Biswas and Ye [3] applied SDP to the problem of sensor localization with noisy range estimates. Subsequently, Kim et al. [14] expanded on this work, exploiting matrix sparsity to reduce the amount of time required to obtain a solution. Recent results by Javanmard and Montanari [12] provide analytical performance bounds for SDP localization where nodes are distributed randomly according to a uniform distribution within a bounded d-dimensional hypercube.

5.4 Graph realization and EBT

In our particular sensor tracking problem, there are two aspects that affect the performance of the realization algorithms. Firstly, the graph is likely to be regularly shaped, as the sensors' movement is sampled from an overlapping trajectory. Secondly, the graph is comprised of several clusters of densely connected vertices (see Fig. 6a). Each cluster corresponds to a single sensor's trajectory, and it is densely connected because is possible to sample an inertial trajectory between any two points. These clusters are loosely interconnected by comparatively fewer edges, which arise from opportunistic contacts. Our first expectation is that DN-SGD will 'push' these clusters away from one other. This will result in there being little spatial overlap between the sensors' trajectories, which will yield an incorrect embedding. Although an iterative refinement step will reduce this error, it will invariably converge to a local minimum. Our second expectation is that MDS will outperform SGD, which we infer directly from the minimization functions. MDS seeks to directly minimize the sum squared error between the observed and predicted distances, whereas SGD seeks to push together those vertices connected by short edges. They key problem is that SGD places no constraint on how close any pair of vertices may be to one another, but only on the spread of the entire embedding. We therefore expect that SGD will yield an acceptable overall distribution of vertices, but on a local level vertices will deviate significantly from their correct position. We expect SDP to perform similarly to MDS, but at a higher computational cost. Finally, we have chosen not to test MDS-MAP(P) for two reasons. Firstly, we expect the graph to be regular, implying that MDS-MAP(P) will perform approximately just as well as straight MDS. Secondly, the computational complexity required to obtain a solution scales poorly with the number of vertices: preliminary experiments showed a run time of 200 seconds for a graph containing fewer than 100 vertices.

6. DRIFT CORRECTION

In the final step of EBT we use the projected graph embedding to drift-correct the sensors' estimated trajectories. Piecewise drift correction is applied independently to each sensors' estimated trajectory between encounter points. We begin with a description of an existing approach from the literature, *Linear Drift Correction*. We then propose a novel alternate method, which we call *Radial Drift Correction*.

Linear Drift Correction was proposed by Constandache et al. [6], and it assumes that the positional estimate drifts linearly with time. Let $\tilde{P}(t)$ be the trajectory estimate for a given node, at some time t. Let t_i and t_{i+1} be the time associated with two adjacent graph vertices representing encounters along a given sensor's path. Let $X(t_i)$ and $X(t_{i+1})$ be the coordinates of the first and second encounter respectively, obtained directly from the graph embedding.

Linear Drift Correction begins by shifting the sensor's estimated trajectory to begin at the position of the first encounter $X(t_i)$. The approach then adds the difference between the estimated and known end points linearly over the period $(t_{i+1} - t_i)$. This is done by first calculating a shift vector $\vec{v} = X(t_i) - \tilde{P}(t_i)$ and then a correction vector $\vec{c} = X(t_{i+1}) + \vec{v} - \tilde{P}(t_{i+1})$. Eqn 6 shows how the final drift-corrected trajectory segment $P(t)$ is calculated.

$$P(t) = \tilde{P}(t) + \vec{v} + \frac{t - t_i}{t_{i+1} - t_i}\vec{c}, \quad t_i \leq t \leq t_{i+1} \qquad (6)$$

We observed that Linear Drift Correction distorted the shape of the curve, most notably in cases where there was a large angular difference in the direction vector between the two points on the estimated trajectory and the two encounter points. To counteract this, we propose an alternate drift-correction method called Radial Drift Correction. In this method the starting point of the estimated trajectory is first shifted to the first encounter. Rather than adding the drift correction linearly along the curve, we rotate the curve about the starting encounter by some angle α, until the estimated displacement vector and actual displacement vector are aligned. Finally, we scale the estimate trajectory by β to align its end point with the second encounter. We denote $\vec{a} = X(t_i) - X(t_{i+1})$ and $\vec{m} = \tilde{P}(t_i) - \tilde{P}(t_{i+1})$ to be the displacement vectors measured between the two given encounter points, according to the embedded graph vertices and the trajectory estimate. Eqn 7 and Eqn 8 show how $\vec{a}$ and $\vec{m}$ are used to find the rotation angle α and scale β.

$$\alpha = atan2(\vec{a}_y, \vec{a}_x) - atan2(\vec{m}_y, \vec{m}_x) \qquad (7)$$

$$\beta = \frac{\|\vec{a}\|}{\|\vec{m}\|} \qquad (8)$$

The updated trajectory segment $P(t)$ is given by Eqn 9 using the rotation angle α and scale factor β defined above.

$$P(t) = X(t_i) + \beta \begin{bmatrix} cos\alpha & -sin\alpha \\ sin\alpha & cos\alpha \end{bmatrix} \left(\tilde{P}(t) - \tilde{P}(t_i)\right) \qquad (9)$$

7. EXPERIMENTAL SETUP

To evaluate the performance of our model we synthetically created five sensors by randomly sampling the source data, discussed in Sec. 7.1.1, at different 30 second intervals. We shifted the all trajectories to begin at time zero, and calculated the encounter points between sensors and anchors. At these encounter points the pairwise distances between sensors were convoluted by additive white noise, with parameters drawn from experiments, which are discussed in Sec. 7.1.2. In Sec. 7.2 we discuss the parameters that were varied, models that were tested and metrics for analysis, while Sec. 7.3 shows how our EBT was calibrated. Each combination of the simulation parameters was repeated 100 times in order to obtain 99% confidence intervals. The simulations were conducted over two days using three parallel Matlab processes on a 2.4 GHz Core 2 Quad with 4GB RAM.

7.1 Source data

7.1.1 *Inertial and ground truth traces*

Our model was evaluated on trace #13 from Angermann et al. [1], which is a recording of a random walk in a 6 x 8 meter room. The trace offers two time-synchronized data streams: a *ground truth* trajectory recorded from an optical localization system, and an *inertial trajectory*, which the authors obtain by passing raw measurements from a shoe-mounted inertial sensor through Foxlin's *Inertial Pedestrian Dead Reckoning* algorithm [9].

7.1.2 *Radio noise model*

To test the performance of our tracking model under more realistic conditions, we used the radio model proposed by Patwari et al [18]. We staggered 10 wireless stations at 1.8m intervals from a given receiver, and measured their signal strength for a 10 minute period. We assumed that the distance d in meters relates to the RSSI r in dBm according to the free space path loss model shown in Eqn. 10. We used a non-linear fitting technique to find p_1 and p_2 for our data set. We assumed further that the received signal strength was corrupted by Additive White Gaussian Noise (AWGN) with a constant standard deviation p_3, measured directly from the trace data. Our radio data and path loss model is summarized in Fig. 10.

$$r = p_1 - p_2 10 log_{10}(d) \qquad (10)$$

We used the following method to model the effect of radio noise on our tracking algorithms. Each simulation run was assigned a radio noise multiplication factor n_f, which weights the standard deviation of the additive white Gaussian noise. Assuming an observed distance of d^t_{ij} between

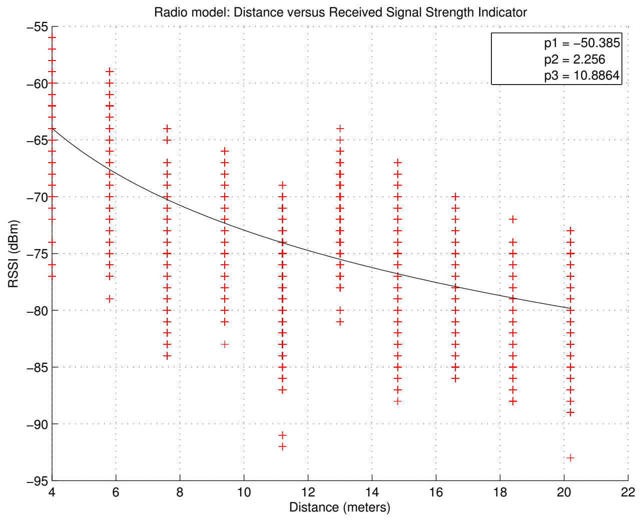


Figure 5: This graph shows the received signal strength (in dBm) as a function of the distance between a sender and receiver. Small crosses represent beacon frame measurements, while the solid line is the path loss model fitted to the measured data.

two sensors i and j at time t, then the RSSI for the beacon frame is given by $\tilde{r}_{ij}^t = p_1 - p_2 10log_{10}(d) + e$, where $e \backsim N(0, n_f p_3)$. This RSSI was fed back through the path loss model to obtain a noise-corrupted distance $\tilde{d}_{ij}^t$.

7.2 Models and Performance Metric

The objective of our experiments was to measure how the following two tracking algorithms perform as a function of the number of anchors (4 to 12), communication range (0.25m to 1.5m) and radio noise factor n_f (0 to 2):

- *Drift Correction Tracking* (DDT) - This model was taken from Constandache et al. [6] and operates in the following way: whenever a sensor encounters another sensor, or anchor, with a fresher positional estimate, it assumes the position of its peer, and corrects for drift between the previous and current encounter.

- *Encounter Based Tracking* (EBT) - This is the proposed tracking algorithm, which was described in Secs 4-6. The model was calibrated as per Sec. 7.3.

We used the Procrustes Transform [7] to express the output trajectories in a common coordinate frame, relative to a minimum of four non-collinear anchor points. The performance of both tracking models was measured by the *root mean square error* (RMSE), over the errors between points along the output trajectory P_i and the ground truth trajectory G_i, for every sensor i. Assuming that we have S sensors and T discrete time ticks, then the RMSE e is given by Eqn 11, where $\|\cdot\|$ represents the standard Euclidean norm.

$$e = \sqrt{\frac{\sum_{s=1}^{S}\sum_{t=1}^{T}\|P_s(t) - G_s(t)\|^2}{ST}} \qquad (11)$$

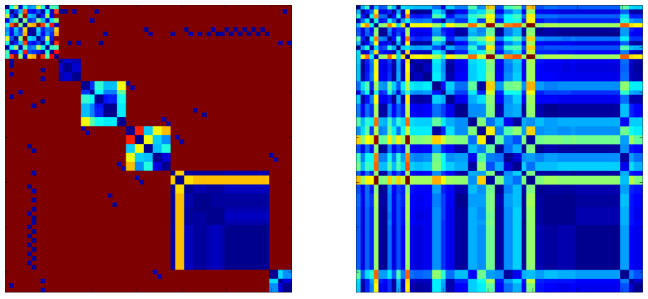


Figure 6: The heat map representation of a sample EBT distance matrix. The upper-left block on the diagonal corresponds to the known anchor distances, while the remaining five blocks correspond to pairwise distances between points on each trajectory. Off-diagonal elements correspond to encounters.

7.3 Calibration of Encounter Based Tracking

7.3.1 *Vertex merging*

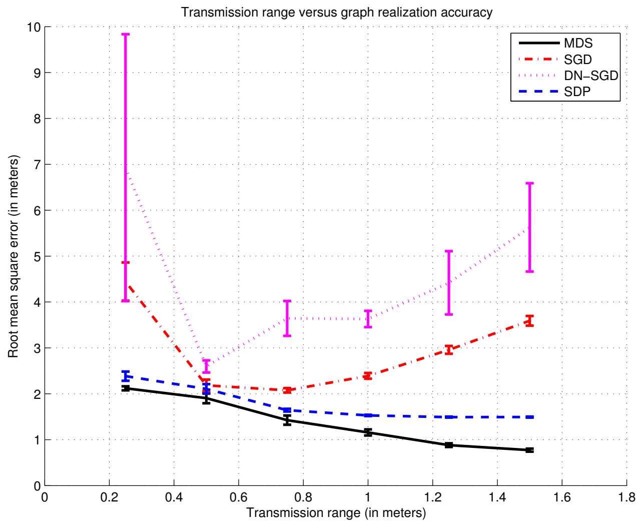
If the vertex merging threshold is set to a value that is too high, then EBT may merge encounters that take place in very different locations. In this case the average distance does not accurately represent the samples that are merged, and the end result is a lower tracking resolution. Conversely, if the value is set too low then the graph contains a large number of vertices, increasing the complexity of graph realization. We determined empirically that five seconds is a sufficient threshold for vertex merging on our trace data set. With this threshold set, in the worst case all graph realization algorithms obtained a solution within one minute.

7.3.2 *Edge selection*

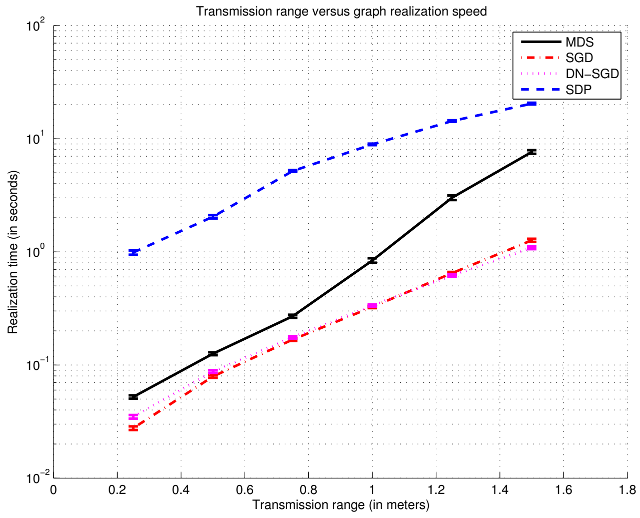
In Sec. 4 we introduced the idea of graph rigidity, and we showed how it relates to unique localizability. We also proposed two edge selection rules for connecting encounters along a single sensor's path. The *complete edge selection rule* fully-connected all encounters on a single sensor's trajectory. The *conservative edge selection rule* sought to reduce the amount of inertial error in the representation, while preserving generic global rigidity. Given perfect distance measurements, it should be possible to find a unique embedding in both cases. However, because of the NP-Hardness of graph realization, in practice this cannot be achieved. Through a series of experiments, we observed that all of the tested approximation algorithms performed favourably using the complete edge selection rule. We attribute this behaviour to the density of the distance matrix and its effect on approximation algorithms for graph realization. Although the remaining edges introduce larger quantities of measurement error, this error has less impact on the final solution that the loss of information about vertex connectivity.

7.3.3 *Graph realization*

Here, our objective was to determine which graph realization algorithm performs well with EBT. To do this we implemented MDS, SGD and DN-SGD, and used the SDP localization implementation by Kim et al. [14]. Fig. 7a and Fig. 7b show how, for these four graph realization al-



(a) Transmission range versus graph realization accuracy.



(b) Transmission range versus graph realization speed.

Figure 7: Comparing the accuracy and speed of EBT using four different graph realization algorithms.

gorithms, EBT accuracy and speed change as a result of an increase in radio transmission range. The RMSE metric used to assess accuracy is given by Eqn 11. An increase in transmission range results in a greater number of encounters, increasing the overall number of vertices in the graph, and thus the speed of graph realization (in our experiments the number of vertices is bounded to approximately 500 by the vertex merging mechanism). As predicted in Sec. 5.4 the accuracy of the two SGD algorithms was significantly lower than the others. SGD, DN-SGD and MDS implementations all rely on eigendecomposition of a square matrix, which means that they should exhibit similar time performances. However, a MDS requires preliminary step to determine unknown elements of the distance matrix (see Fig. 6). Our implementation uses a Floyd-Warshall shortest path algorithm, which has a complexity of $O(n^3)$ for a matrix with $n \times n$ vertices. This is why the speed of MDS is lower than the other two methods. Our results suggest that for problems with over 500 encounters, switching from MDS to SDP may provide a good trade-off between speed and accuracy.

8. RESULTS AND DISCUSSION

Fig 8 provides a summary of our sensitivity analysis. We have included an *Inertial* curve, which shows the RMSE of the inertial data without any drift-correction. It serves as a worst-case benchmark for the tracking algorithms. Take note that only noise-free data is used in Figs 8a and 8b. In EBT, Radial Drift Correction showed a consistent 10cm improvement over Linear Drift Correction.

8.1 Sensitivity analysis for EBT and DDT

8.1.1 Number of anchors

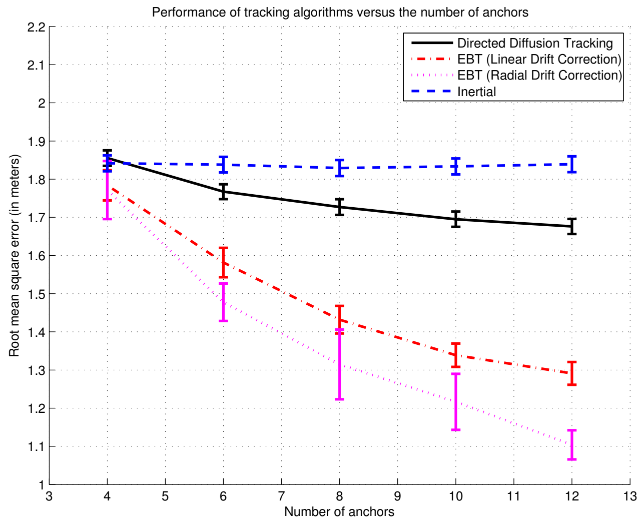
For DDT, increasing the number of anchors is equivalent to sampling the search space at a greater resolution. As Fig. 8a illustrates, the DDT error drops at a decreasing rate, as more anchors are added to the system. Presumably, this is because a doubling in resolution requires a square increase in the number of anchors. For EBT, we observe that an increase in the number of anchors greatly improves the tracking performance. Increasing the number of anchors yields more encounters, thereby increasing the total number of sensor to anchor encounters in the experiment. In addition to adding more information to the model, such encounters help 'pin' the sensors' trajectories to the rigid cluster formed by the anchors, thereby improving graph rigidity.

8.1.2 Transmission range

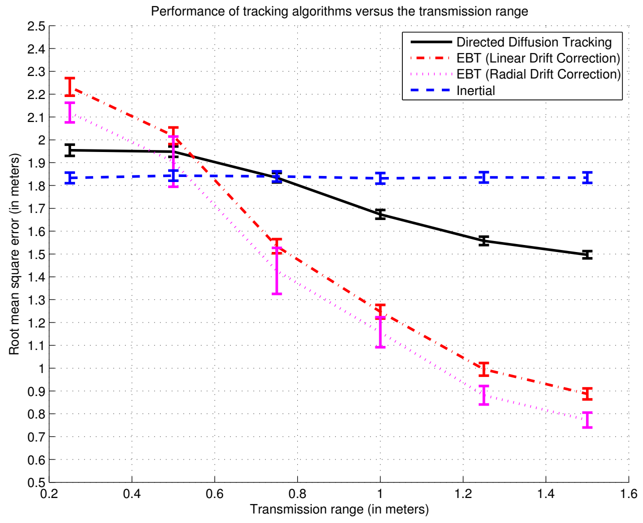
Fig. 8b shows how the performance of DDT and EBT change as the radio transmission range increases, equally for both sensors and anchors. In general, an increase in the transmission range yields a greater number of encounters. This adds more information to both EBT and DDT, thus improving performance. A transmission range of less than 0.5m yields very few encounters. For DDT this means that positional updates are less frequent, yielding a poor tracking performance. For EBT this poses a bigger problem – a small number of encounters causes a flexible encounter graph, resulting in a bad realization (see Sec. 8.2). This is why the EBT error for small transmission ranges is greater than DDT. However, once the graph is rigid, EBT performs significantly better than DDT. We attribute this to two things. Firstly, DDT assumes that when an encounter takes place, the two devices are collocated. As the transmission range increases, so this assumption becomes less valid. Secondly, EBT performs a global correction of encounter points, whereas DDT perform a local correction that only involves the two sensors at the point of encounter.

8.1.3 Transmission noise

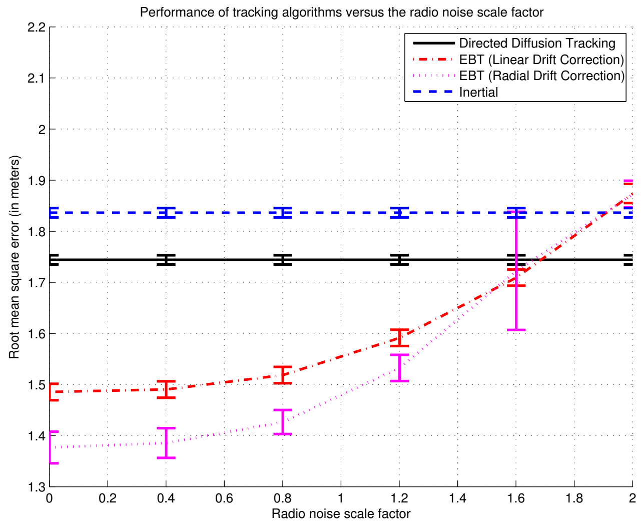
Fig. 8c shows how the performance of EBT and DDT scale as more white noise is added to radio distance estimates. Since DDT does not take into account radio distance, it follows that its performance remains consistent across all experiments. EBT showed an exponential drop in performance as a function of noise. This relationship arises directly a result of the path loss model: a linear increase in RSSI error



(a) Increasing the number of fixed anchors.



(b) Increasing the transmission range.



(c) Increasing the RSSI additive white noise variance.

Figure 8: Sensitivity analysis results for Encounter Based Tracking and Directed Diffusion Tracking.

yields an exponential error in distance. Extremely noisy distance data results in EBT pushing apart sensor's subgraphs in the plane, in order to satisfy artificially-long edges.

8.2 The effect of graph rigidity on EBT

One limitation of EBT is that it requires that the underlying graph is globally rigid. Although we can construct globally rigid subgraphs for each sensor, the global rigidity of the joint graph is a function of the opportunistic encounters made by each sensor. If a graph does not satisfy global rigidity, multiple embeddings will exist that satisfy the observed pairwise distances. Fig. 9a shows a graph from an example simulation run that exhibits such behaviour. We analysed the graph with the 2D Pebble Game [11] in order to find the minimum number of locally rigid clusters. Local flexibility results in unconstrained vertices, which causes an indeterminacy in the realization solution space. This manifests in a poor embedding, resulting in the trajectory being reconstructed incorrectly, which we highlight in Fig. 9b.

9. CONCLUSION AND FUTURE WORK

In this work we proposed a novel offline Encounter Based Tracking (EBT) algorithm, which uses opportunistic peer-to-peer and sensor-to-anchor radio encounters to correct drift-corrupted trajectories. Although EBT may also be used for three dimensional tracking, polynomial time rigidity certification for three dimensions and higher remains an open problem. Through experimentation, we have shown that 2D EBT tracking outperforms a competing model in the literature, over a wide range of network parameters. Our results also show that EBT scales predictably with additive white Gaussian noise corrupted radio distances. In future work we plan to investigate more sophisticated drift-correction algorithms, how to include measurement uncertainty into EBT, and how to calibrate EBT for scenarios containing irregular graph formations – those in which the encounter points are not uniformly distributed on the plane.

Acknowledgements

This research was supported by the Sensing Unmanned Autonomous Aerial Vehicles (SUAAVE) research project under EPSRC grant EP/F064217/1. We would like to thank Hongkai Wen for his help collecting RSSI data, and Simon Julier, Mihai Cucuringu, Andrew Markham, Sarfraz Nawaz and Nadine Levin for reviewing earlier versions of this work.

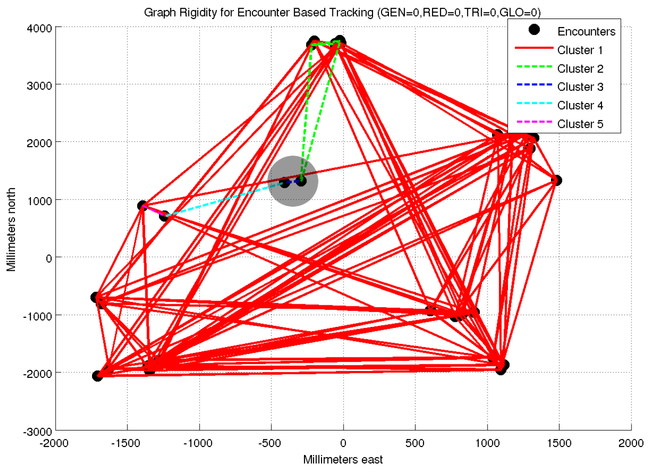
10. REFERENCES

[1] Angermann, M., Robertson, P., Kemptner, T., and Khider, M. A high precision reference data set for pedestrian navigation using foot-mounted inertial sensors. In *Indoor Positioning and Indoor Navigation (IPIN), 2010 Intl Conference on* (2010), pp. 1–6.

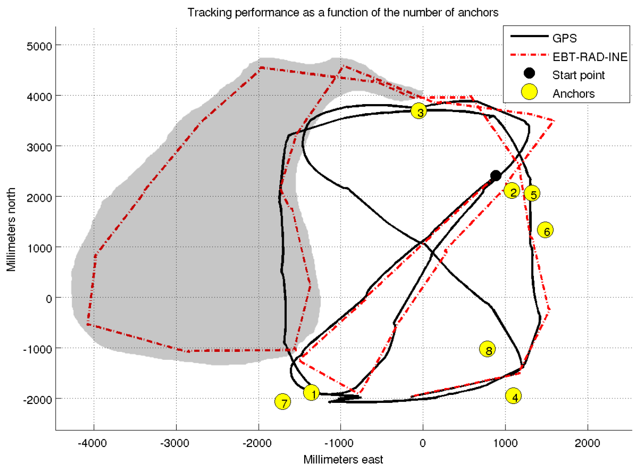
[2] Aspnes, J., Goldenberg, D., and Yang, Y. R. On the computational complexity of sensor network localization. *Algorithmic Aspects of Wireless Sensor Networks* (2004), 32–44.

[3] Biswas, P., and Ye, Y. Semidefinite programming for ad hoc wireless sensor network localization. In *Proc. of the 3rd Intl Symposium on Information Processing in Sensor Networks* (2004), pp. 46–54.

[4] Broxton, M., Lifton, J., and Paradiso, J. A. Localization on the pushpin computing sensor network



(a) Rigidity graph highlighting two non-rigid vertices.



(b) The net effect of non-rigid vertices on sensor tracking.

Figure 9: The effect of non-rigid graph vertices on the accuracy of Encounter Based Tracking.

using spectral graph drawing and mesh relaxation. *ACM SIGMOBILE Mobile Computing and Communications Review 10*, 1 (2006), 1–12.

[5] Cabero, J. M., Torre, F. D. l., Sanchez, A., and Arizaga, I. Indoor people tracking based on dynamic weighted multidimensional scaling. In *MSWiM '07. Proc. of the 10th ACM Symposium on Modeling, Analysis, and Simulation of Wireless and Mobile Systems* (2007), pp. 328–335.

[6] Constandache, I., Bao, X., Azizyan, M., and Choudhury, R. R. Did you see Bob?: human localization using mobile phones. In *Proceedings of the sixteenth annual international conference on Mobile computing and networking* (2010), pp. 149–160.

[7] Cox, M., and Cox, T. Multidimensional scaling. *Handbook of data visualization* (2008), 315–347.

[8] Eren, T., Goldenberg, O. K., Whiteley, W., Yang, Y. R., Morse, A. S., Anderson, B. D. O., and Belhumeur, P. N. Rigidity, computation, and randomization in network localization. In *INFOCOM 2004. 23rd Conf. of the IEEE Computer and Comms Societies* (2004), vol. 4, pp. 2673–2684.

[9] Foxlin, E. Pedestrian tracking with shoe-mounted inertial sensors. *IEEE Computer Graphics and Applications* (2005), 38–46.

[10] Jackson, B., and Jordán, T. Connected rigidity matroids and unique realizations of graphs. *Journal of Combinatorial Theory, Series B 94*, 1 (2005), 1–29.

[11] Jacobs, D. J., Thorpe, M. F., and Chubynsky, M. 2D Pebble Game with Central Forces, November 2011.

[12] Javanmard, A., and Montanari, A. Localization from incomplete noisy distance measurements. In *Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on* (31 2011-aug. 5 2011), pp. 1584–1588.

[13] Kim, B., Kaess, M., Fletcher, L., Leonard, J., Bachrach, A., Roy, N., and Teller, S. Multiple relative pose graphs for robust cooperative mapping. In *Robotics and Automation (ICRA), 2010 IEEE Intl Conf. on* (may 2010), pp. 3185–3192.

[14] Kim, S., Kojima, M., and Waki, H. Exploiting sparsity in SDP relaxation for sensor network localization. *Tokyo, Japan: Department of Mathematical and Computing Sciences, Tokyo Institute of Technology* (2008).

[15] Koren, Y. On spectral graph drawing. *Computing and Combinatorics* (2003), 496–508.

[16] Laman, G. On graphs and rigidity of plane skeletal structures. *Journal of Engineering mathematics 4*, 4 (1970), 331–340.

[17] Macagnano, D., and de Abreu, G. Tracking multiple targets with multidimensional scaling. In *Wireless Personal Multimedia Comms* (2006).

[18] Patwari, N., Ash, J. N., Kyperountas, S., Hero, A. O., . I. I. I., Moses, R. L., and Correal, N. S. Locating the nodes: cooperative localization in wireless sensor networks. *Signal Processing Magazine, IEEE 22*, 4 (july 2005), 54–69.

[19] Saxe, J. B. *Embeddability of weighted graphs in k-space is strongly NP-hard*. Carnegie-Mellon University, Dept. of Computer Science, 1980.

[20] Shang, Y., and Ruml, W. Improved MDS-based localization. In *INFOCOM 2004. 23rd Conf. of the IEEE Computer and Communications Societies* (2004), vol. 4, pp. 2640–2651.

[21] Shang, Y., Ruml, W., Zhang, Y., and Fromherz, M. P. J. Localization from mere connectivity. In *Proc. of the 4th ACM international symposium on Mobile ad hoc networking & computing* (2003), pp. 201–212.

[22] Thrun, S., and Montemerlo, M. The graph SLAM algorithm with applications to large-scale mapping of urban structures. *The International Journal of Robotics Research 25*, 5-6 (2006), 403–429.

[23] Wymeersch, H., Lien, J., and Win, M. Z. Cooperative Localization in Wireless Networks. *Proceedings of the IEEE 97*, 2 (feb. 2009), 427–450.

[24] Zhu, Z., So, A. M.-C., and Ye, Y. Universal Rigidity and Edge Sparsification for Sensor Network Localization. *SIAM J. on Optimization 20*, 6 (oct 2010), 3059–3081.